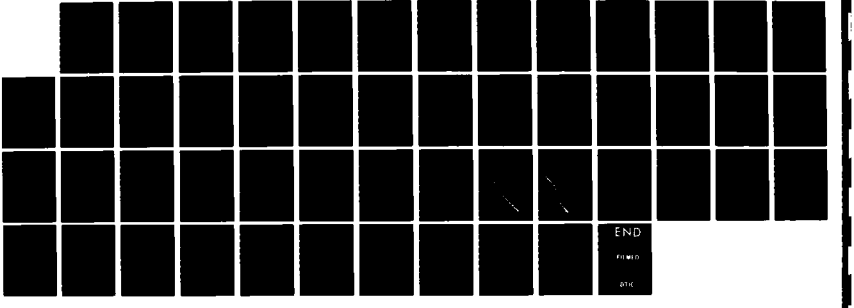
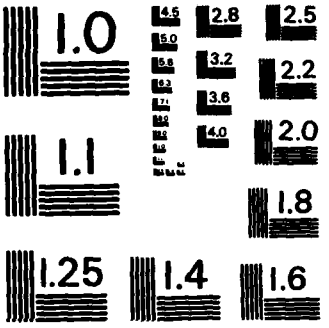
AD-A160 205 MEMORY CONFLICT SIMULATION OF A MANY-PROCESSOR CRAY ARCHITECTURE PART 1 A..(U) MICHIGAN UNIV ANN ARBOR SUPERCOMPUTER ALGORITHM RESEARCH LAB.. 1/1

UNCLASSIFIED D A CALAHAN ET AL. 01 MAR 85 SARL-6 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Report SARL #6

Memory Conflict Simulation of a Many-Processor CRAY Architecture. Part I: A CRAY X-MP Study

D. A. Calahan

Ken Elliott, III

March 1, 1985

Sponsored by
Los Alamos National Laboratory
Air Force Office of Scientific Research

Supercomputer Algorithm Research Laboratory
Department of Electrical Engineering & Computer Science

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

AD-A160 205

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS:
2a. SECURITY CLASSIFICATION AUTHORITY: ---	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S):	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR-85-0765
6a. NAME OF PERFORMING ORGANIZATION: University of Michigan; 6b. OFFICE SYMBOL (If applicable):	7a. NAME OF MONITORING ORGANIZATION: AFOSR
6c. ADDRESS (City, State and ZIP Code): Dept. of Elec. Eng. and Computer Science, Ann Arbor, MI 48109	7b. ADDRESS (City, State and ZIP Code): Bldg. 410, Bolling AFB, D.C. 20332
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-84-0096
8c. ADDRESS (City, State and ZIP Code): Bldg. 410, Bolling AFB, D.C. 20332-6448	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2304	A3	

11. TITLE (Include Security Classification): Memory Conflict Simulation of a Many-Processor CRAY Architecture. Part I: A Cray X-MP Study

12. PERSONAL AUTHOR(S): D. A. Calahan and K. B. Ellitoo, III

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Interim	FROM ____ TO ____	1 March 1985	45

16. SUPPLEMENTARY NOTATION:

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Linear algebra, Supercomputers, Computer memories
XXXXX	XXXXXXXXXXXXX		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The performance of three Fortran kernels and two assembly kernels (including two linear algebra kernels) is simulated for a CRAY X-MP multiprocessor architecture of up to 16 processors and 256 memory banks. The effects of variations on the X-MP-2 memory conflict resolution protocol, including X-MP-4 protocol, are studied.

Solution 2 keywords: Supercomputers; Simulators.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☒ DTIC USERS ☐	Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
John P. Thomas, Jr., Capt, USAF	(202)767-5026	NM

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

Memory Conflict Simulation of a Many-Processor CRAY Architecture. Part I: A CRAY X-MP Study

D. A. Calahan

Ken Elliott, III

March 1, 1985

Supported by

Los Alamos National Laboratory

and

Air Force Office of Scientific Research

Under Grant 84-0096

Abstract

The performance of three Fortran kernels and two CAL kernels is simulated for a CRAY X-MP architecture of up to 16 processors and 256 memory banks. The effects of variations on the X-MP-2 memory conflict resolution protocol, including X-MP-4 protocol, are studied.

Acknowledgements

The cooperation of Chris Hsiung and Al Schiffleger of Cray Research, Inc., was indispensible in resolving intricacies of the XMP memory system. The assistance of Jim Arnold and Ken Stevens of NASA/ARC and of Carl Diem of CRI in providing XMP access is also acknowledged.

Simulator Availability

The XMP simulator was developed using private resources and is proprietary to Professor Calahan. Because it contains information restricted by CRI, access is also restricted.

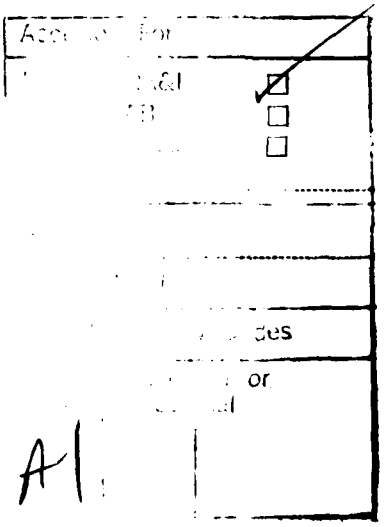


TABLE OF CONTENTS

PAGE

I. INTRODUCTION 1

II. HARDWARE REVIEW 2

A. INTRODUCTION 2
B. X-MP-2 CONFLICT RESOLUTION 2
C. X-MP-2M CONFLICT RESOLUTION 4

III. SIMULATION STUDIES 5

A. THE SIMULATOR 5
B. THE EXPERIMENTS 5
C. DELAY DEFINITION AND SIMULATOR 5
D. DELAY DISTRIBUTION FUNCTIONS 9
E. ORIGINS OF ACCESS DELAY 9
1. Introduction 9
2. Effect of Buffer Fetchs 13
3. Section Conflicts 14
a. Introduction 14
b. Steady state section conflicts 17
c. Two-port startup section conflicts 21
d. Three-port startup section conflicts 22
4. Effects of Section Design Parameters 25
5. Startup Bank Conflicts 28

IV. EVALUATION OF CONFLICT RESOLUTION PROTOCOLS 34

A. INTRODUCTION 34
B. AGGRAGATE PERFORMANCE 34
C. DEPENDENCE OF D_{ac} ON R_{bp} 37

REFERENCES 39

APPENDIX A. EXPERIMENT DESCRIPTION 40
APPENDIX B. SIMULATOR VALIDATION 43

I. INTRODUCTION

Research involving efficient use of commercial multiprocessor scientific architectures such as the CRAY X-MP is presently focused on algorithmic decomposition of problems into large concurrent tasks. Early evidence indicates that if the number of processors (=p) is small (say $2 \leq p \leq 16$) many problems can be decomposed into such large tasks that the speedup achieved is nearly equal to p [1][2]

At this high efficiency, a heretofore second-order effect may begin to develop importance, namely, the interference of reads and writes attempting to simultaneously access shared memory resources (memory banks, sections, etc). As p increases from the present 2 and 4 to 8, 16, and beyond, increasing the number of banks correspondingly not only increases the read/write time - for conflict checking - but also imposes severe problems in high-speed chip and memory organization.

Recent related studies have examined this problem with real codes and a generic class of processors [3], and for the CRAY X-MP [5] with random memory fetches to gain insight into the effects of various conflict protocols on access delays.

In this report, the mechanisms which account for the delay of memory accesses is studied with the aid of an instruction-level timing simulator for the CRAY X-MP family of processors. The accesses associated with running Fortran and assembly codes on an MP of up to 16 processors are studied using simulator instrumentaion which records delay and other information.

Projections are made which indicate that the X-MP-4 conflict resolution protocol, if used with 8 and 16 processors, creates significantly longer access delays than the X-MP-2 protocol.

In a companion report [8], the separate question of the effects of access delays on algorithms is studied, and conflict-resistant algorithms are proposed.

II. HARDWARE REVIEW

A. INTRODUCTION

Although certain definitions and observations may be appropriate to other classes of multiprocessors, this study is most directed at vector multiprocessors such as the CRAY X-MP where shared memory access rates are high for typical scientific vector codes. Indeed the motivation for this study requires some knowledge of the X-MP organization and operation. This will be reviewed below; more related discussion is given in [5] and [6].

B. X-MP-2 MEMORY AND CONFLICT RESOLUTION

Figure 1 shows the shared memory organization of the X-MP-2, the two processor X-MP extended to p processors in the simulator. For each processor, every fourth bank is accessed through the same section; with p processors, there are 4 p sections. Conflicts occur at the bank or section level, as follows:

> Bank-Busy conflict - The Bank Busy conflict is caused by any port within or between CPUs requesting a bank currently in a reference cycle. Resolution of this conflict occurs when the bank cycle is complete. Hold reference because of a Bank Busy conflict is 1, 2, or 3 CPs.
>
> Simultaneous Bank conflict - The Simultaneous Bank conflict is caused by two or more ports in different CPUs requesting the same bank. Resolution of this conflict is based on a priority (see below). Hold reference is a 1 CP because of a Simultaneous Bank conflict. A Bank Busy conflict always follows a Simultaneous Bank conflict.
>
> Section conflict - The Section conflict is caused by two or more ports in the same CPU requesting any bank in the same section. Resolution of this conflict is based on a priority, the Bank Busy conflict, and Simultaneous Bank conflict. The highest priority port with no Bank Busy conflict and no Simultaneous Bank conflict is allowed to proceed, all other ports involved in this conflict hold (see below). Hold reference is 1 CP because of a Section conflict.

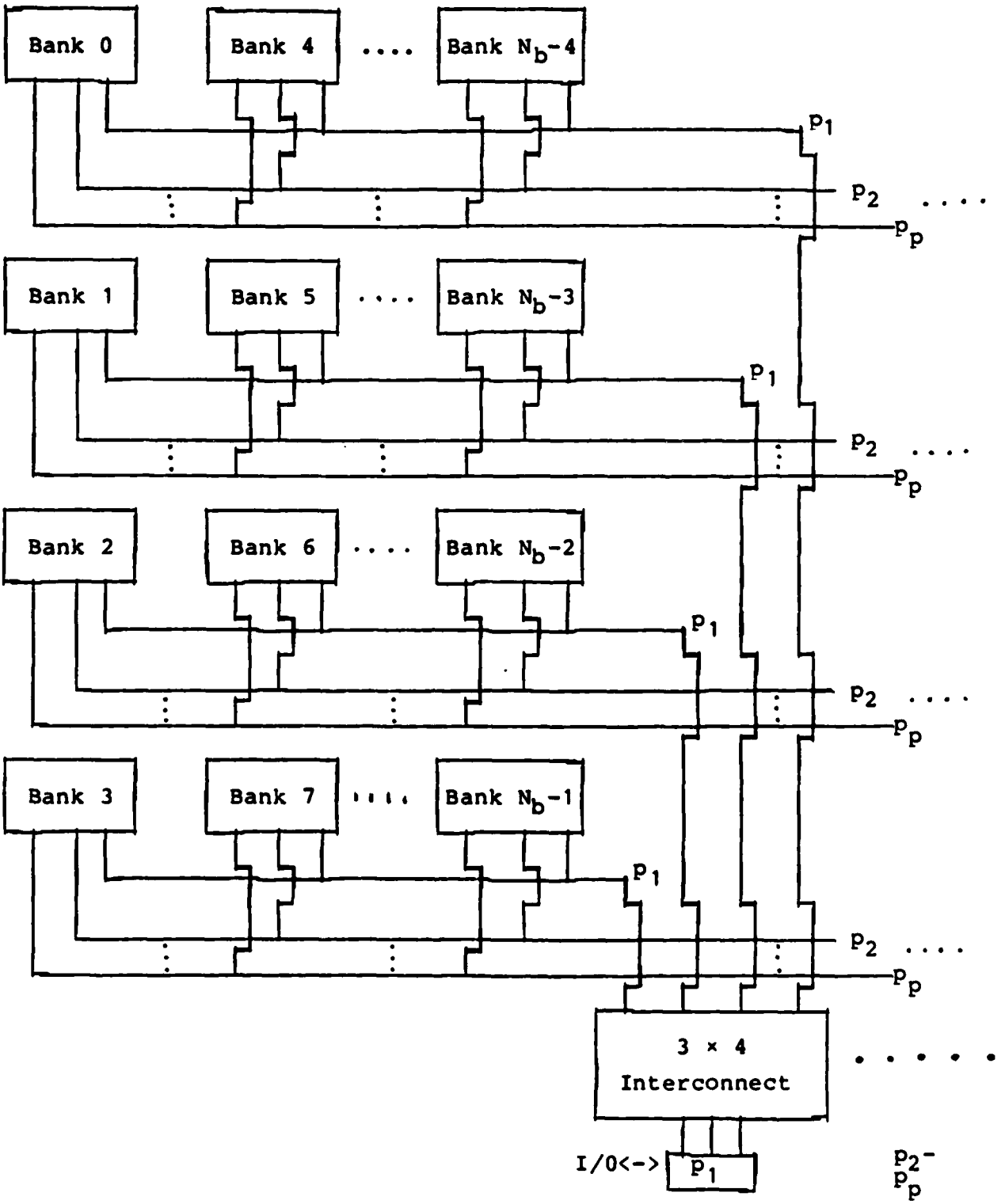


Figure 1. Simulated CRAY X-MP memory organization

When these rules fail to resolve a conflict, the vector stride and instruction issue time are utilized to establish priority [6].

Extensions of the buffer fetch reservation protocol to 16 processors are discussed later.

C. X-MP-2M CONFLICT RESOLUTION

The X-MP-4 resolution protocol has two major differences from the X-MP-2. To evaluate the effects of one of these, an intermediate (hypothetical) processor has been simulated, the X-MP-2M (M for modified). The second difference will be studied in a future report.

In the X-MP-2M, the sections are numbered differently from the X-MP-2. In the X-MP-2, bank #b is in section number s = mod(b,4). In the X-MP-2M, bank #b is in section number s = mod(b/4,4). This groups banks in fours, where each group of 4 belongs to one of four sections. It has been shown independently in [5] that this avoids certain catastrophic conflict patterns associated with the X-MP-2.

III. SIMULATION STUDIES

A. THE SIMULATOR

An instruction-level simulator produces numerical and timing information for the X-MP-2, the X-MP-2M, and the X-MP-4. The general timing accuracy of the X-MP-2 simulator is .2% for a uniprocessor and 1.3% for 2-processor hardware. Codes of read and write instructions only are exact with conflicts. Documentation of this concurrence is given in Appendix B. The X-MP-4 simulator does not incorporate several minor timing differences of the X-MP-4 hardware vis-a-vis the X-MP-2; its timing accuracy has not been validated at this writing.

B. THE EXPERIMENTS

A number of parameters were involved in this simulation study.

(a) Codes ranged over three Fortran-derived and two CAL codes (see Appendix A).

(b) Processors ranged from 1 to 16.

(c) Banks ranged from 16 to 256; the ratio

$$R_{bp} = \frac{\text{\# of banks}}{\text{\# of processors}} \qquad (1)$$

ranged from 4 to 64.

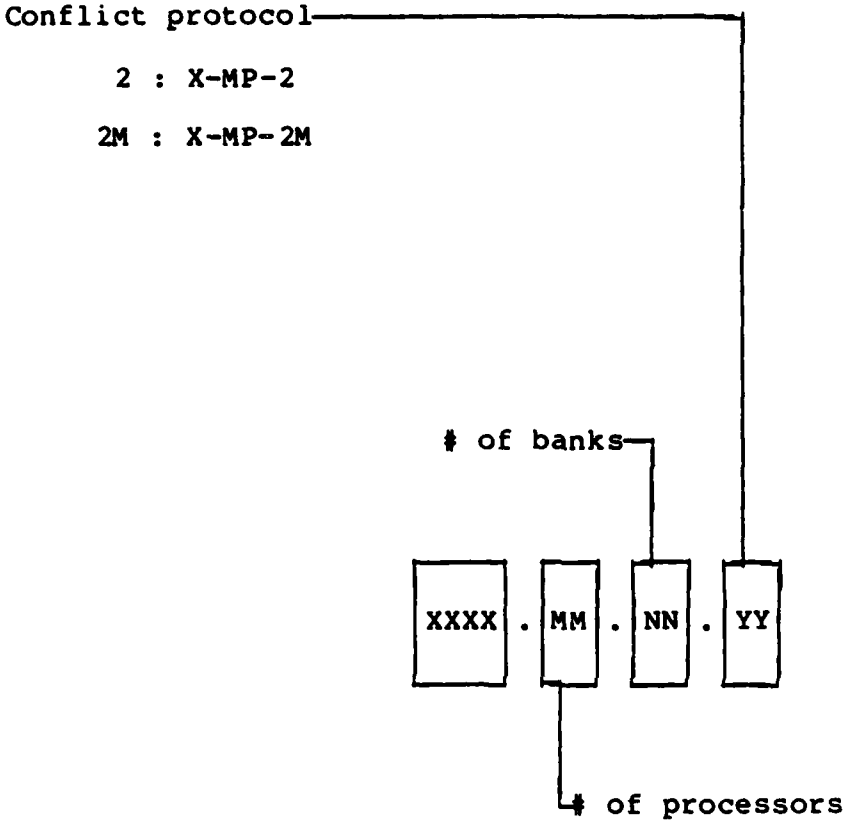
(d) Bank conflict protocols of the X-MP-2, X-MP-2M, and X-MP-4 were studied.

The runs will be indicated by the nomenclature of Table 1.

In these experiments, nearly all vectors had a length (VL) of 64 and a stride of unity. This will be assumed in all analyses.

C. DELAY DEFINITION AND SIMULATION

A unit stride vector access is depicted in the memory utilization map of Figure 2, which displays the reservations



Code

MUL1: medium access Fortran matrix multiply

MUL2: high access Fortran matrix multiply

MUL3: low access CAL matrix multiply

RAN : high access random CAL read/write only

FFT : multiple (64) 8-point FFT's

CFD : fluid dynamics kernel

Table 1. Experiment designations

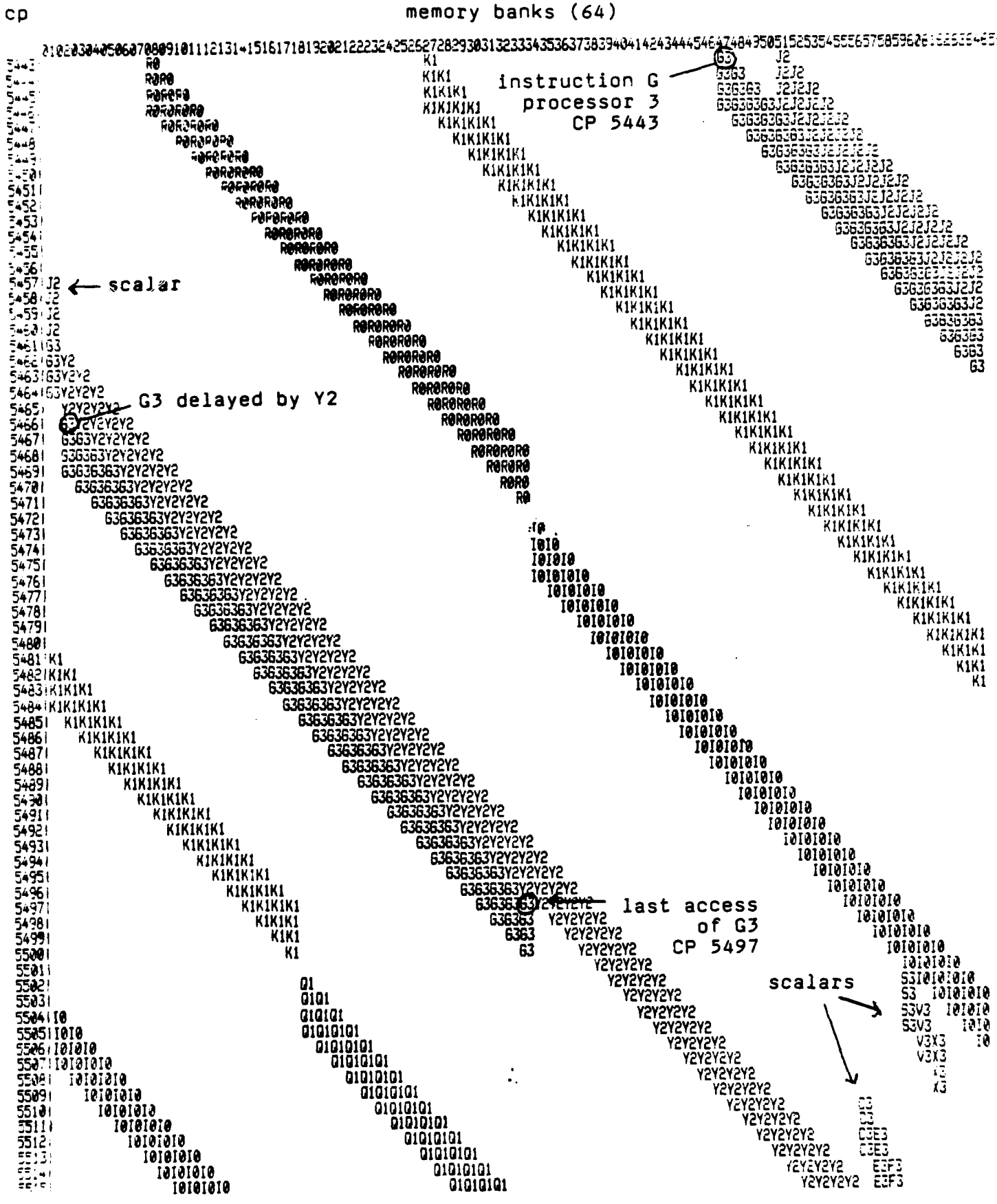


Figure 2. Memory utilization map

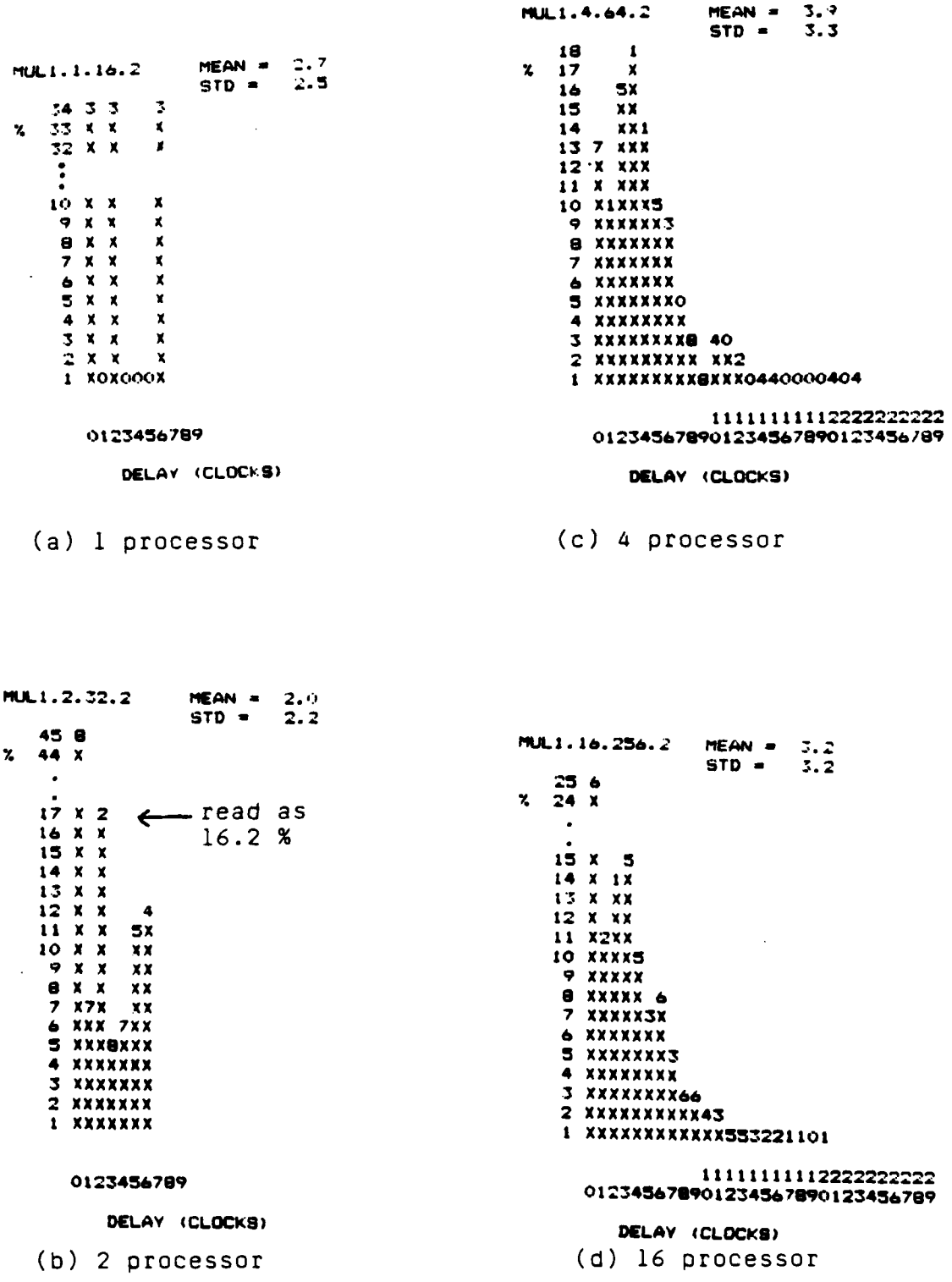


Figure 3. Display of delay granularity as a function of the number of processors.

Figure 4a. Memory utilization map with buffer fetches

placed on memory banks at each CP. The first element of a 50-length vector is accessed in memory at CP = 5443 and reserves the bank for 4 CP's. The last element is accessed at CP = 5497.

Define

T_F = time of attempted access of first vector element

T_L = time of access of last vector element

where times are measured in CP's. Then define the access delay

$$D_{ac} = T_F - T_L - VL + 1$$

D_{ac} is equal to 4 for the above case. The same definition applies to vector reads and writes.

Simulation of only the conflict resolution protocol of any of the CRAY family of processors is straightforward for sequences of memory read and write instructions only. One can then test the statistical variation of access delay with reads and writes distributed (for example) uniformly across the banks [5]. However, implementation of such protocol in a general instruction-level simulator is more complicated. Among other problems, one must accommodate for the effects of access delays on instructions involving these accesses as operands or results. Thus, an entire "chain" of instructions must be held if a read or write is involved.

Two advantages result from implementation in a general simulator.

(a) The access delays associated with actual codes can be determined; not only may these have nonuniform bank access distributions but (static) memory utilization will vary with the code (see Appendix A).

(b) Algorithm delay - the timing overhead in total code execution resulting from access delays - can be

measured. This is discussed in [8], where it is shown that, although access delay and algorithm delay may apparently be related by rules of thumb, algorithms can be designed to have a low sensitivity to even large access delays.

D. DELAY DISTRIBUTION FUNCTIONS

The simulator can be instrumented to record D_{ac} for each access. Over an entire simulation, these can be normalized as distribution functions, such that the functional value for each delay is the fraction of its occurrence relative to all accesses, expressed as a %. This will be termed a DDF (delay distribution function).

Among other uses of DDF's to be studied below, they can demonstrate the granularity of the values a delay may assume, as a function of the number of processors. In Figure 3a, for example, a uniprocessor matrix multiply incurs only three delays (0, 2, and 6 CP). However, as the number of processors increases, a much smoother DDF is observed.

E. ORIGINS OF ACCESS DELAY

1. Introduction

In the process of simulating hundreds of combinations of codes, conflict protocols, and numbers of banks and processors, examples have been found that accent the major sources of delay. These will be illustrated with the simulator instrumentation. Their significance to individual runs will also be evaluated; their importance to aggragate performance will be considered in Section IV.

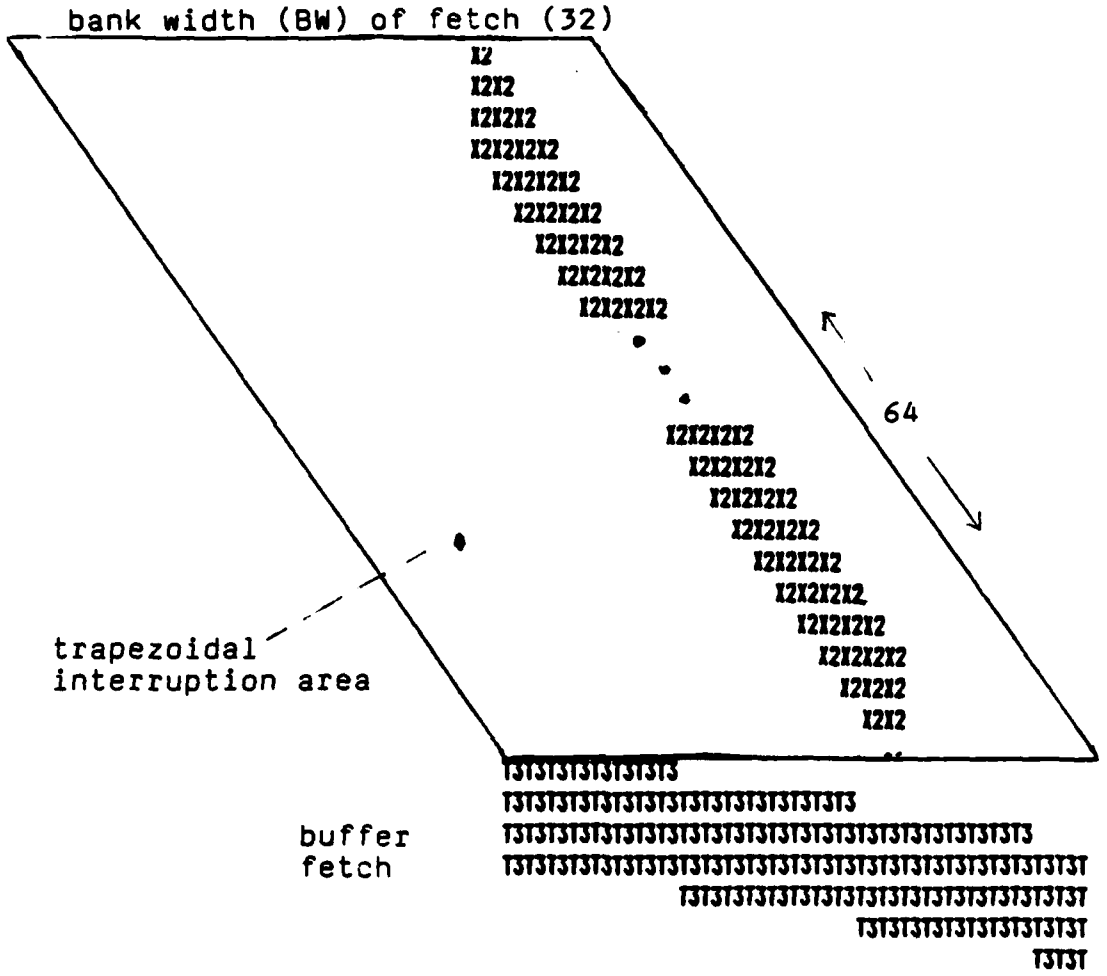
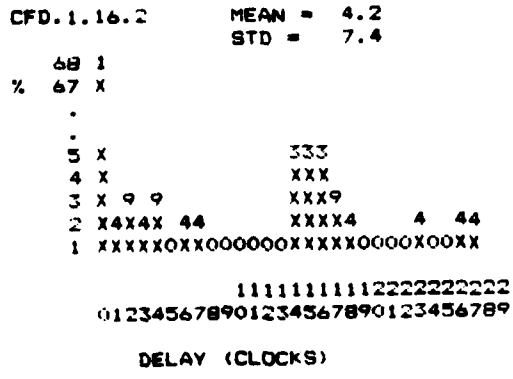
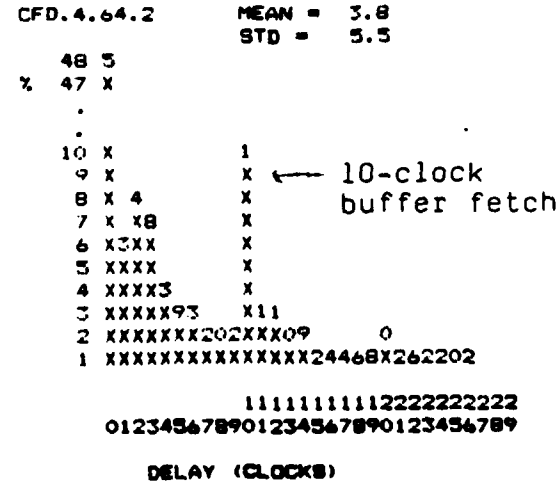


Figure 4b. Buffer fetch analysis



(a) 16 bank



(b) 64 bank

Figure 5. Illustration of buffer fetch delays

2. Effect of buffer fetches

In extending the X-MP-2 to more processors, the instruction buffer fetch protocol of the X-MP-4 is adapted. Here, the processors are "paired", so 8 data ports are available for a fetch. A reservation is placed on all 32 banks that will be referenced during the fetch, and all references made by the CPU's that form the "pair" are held. The references for all other CPU's, as long as they do not access the banks used for the fetch, are allowed to proceed. Two examples are shown in Figure 4a. The reservation on the banks is for 7 CP (until all banks involved in the fetch have cycled). This results in a 10-clock minimum delay in any access interrupted by a buffer fetch. With 16 banks, the lowest number simulated, pairing is not possible and a 14-clock delay results. The DDF's for CFD.4.64.2 and CFD.1.16.2 simulations are shown in Figure 5. The 10-clock delay peak is clearly shown in Figure 5b; in Figure 5a, the delays increase abruptly at 14 clocks.

From knowledge of the number of banks (NB), the bank-width (BW) of an instruction fetch, and the number of fetches (NF) over the execution time (T) of a run, an estimate may be made of the fraction (f_I) of interrupted vector accesses.

A single interruption will occur if a vector is initiated, without interference from other vectors, in the trapezoidal area illustrated in Figure 4b. This area is (BW)(VL-1) bank-clocks; the total area of hazard with NF fetches in (NF)(BW)(VL-1), from a area of (NB)(T) bank-clocks for the total simulation. The fraction of vectors interrupted by a buffer fetch is therefore

$$f_I = \frac{(NF)(BW)(VL)}{(NB)(T)}$$

$$= \frac{(P)(NFP)(BW)(VL)}{(NB)(T)}$$

$$= \frac{(NFP)(BW)(VL)}{(R_{bp})(T)} \qquad (2)$$

where P is the number of processors, NFP is the number of buffer fetches per processor and R_{bp} is the ratio of banks to processors. For the CFD code, NFP = 6 and T = 6413 clocks; with BW = 32, VL = 64, and R_{bp} = 16,

$$f_I = \frac{(6)(32)(64)}{(16)(6413)}$$

$$= .12 \qquad (3)$$

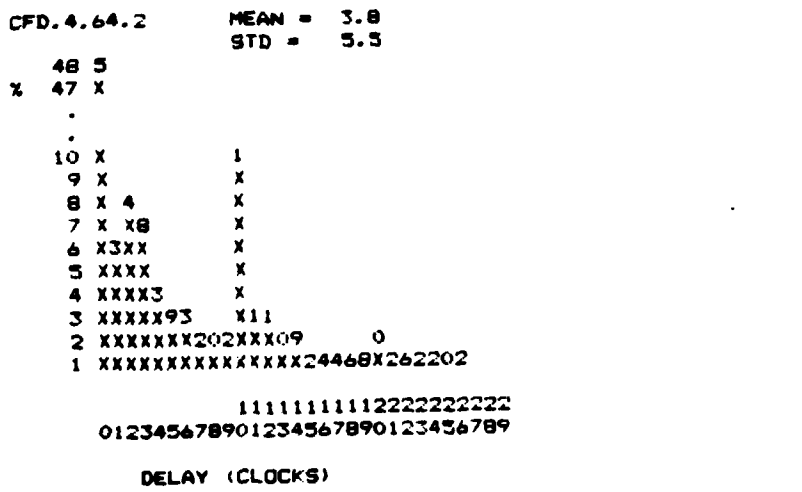
This fraction is supported by the DDF's of the associated CFD code of Figure 6, where the ratios of (10-clock delays)/(0-clock delays) are .19, .14, and .15 for p = 4,8 and 16 processors respectively. These latter ratios will normally be higher than predicted by Eq. (2), since some 10-clock delays will occur even without buffer fetches. The DDF's of Figure 6 show a small increase for 20-clock delays as well, indicating the fraction of reads which encounter two buffer fetches.

The average access delay introduced by these fetches ($\bar{D}_{acbf}$) is simply $10f_I$ or 1.2 clocks in the above case.

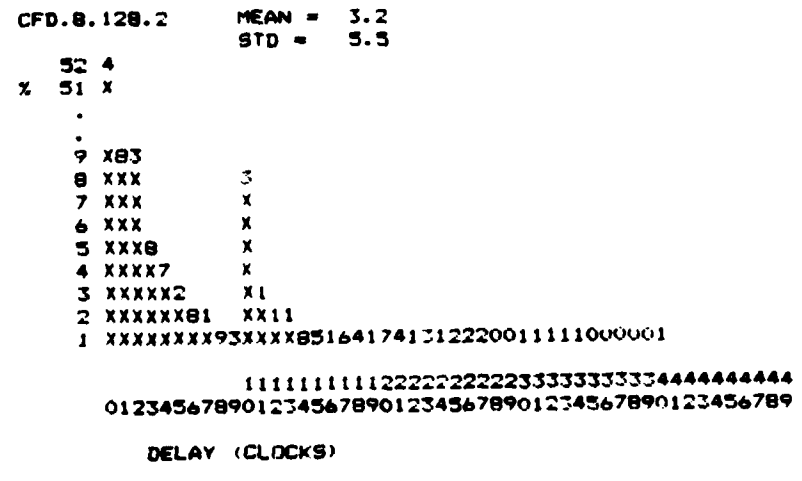
3. Section conflicts

a. Introduction

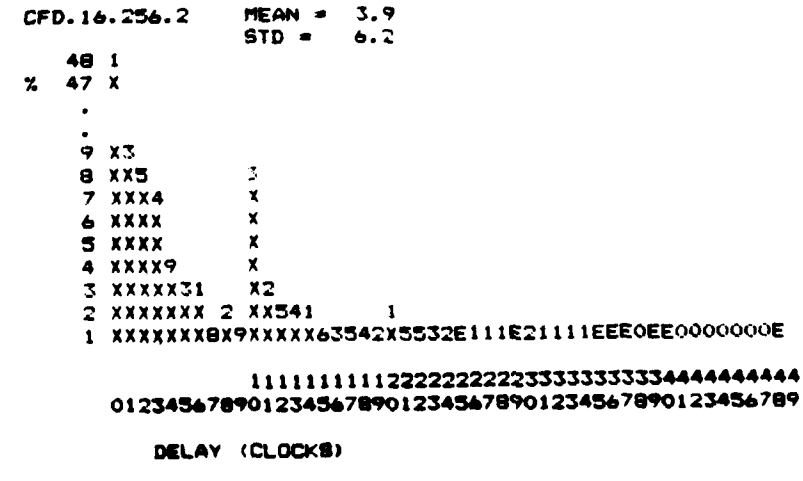
In a code executing from a processor with more than one



(a) 4 processor



(b) 8 processor



(c) 16 processor

Figure 6. DDF's for 4,8, and 16-processor executions of CFD code, with R_{bp} = 16.

active port, there is a potential for conflicts between ports at the section level. These occur either as (a) steady-state conflicts or (b) startup conflicts.

Among the factors likely to influence the number of section conflicts is the number of active ports. A code which uses only one active port may suffer bank conflicts, but not section conflicts; this may result in less access delay than a code with less total memory traffic shared between two or three ports.

The simulator has a capability to monitor dynamically the number of simultaneous accesses from each processor. The fractions of total run times that 0, 1, 2, and 3 ports are busy n a typical processor is given in Appendix A.

b. Steady-state section conflicts

It has been observed in [5] that in the X-MP-2, a steady state section conflict between accesses from a CPU can occur when both ports vie for neighboring memory banks. An example is shown in Figure 7 between instructions A3 and Q3, where every fourth access is delayed by one clock. This can create a worst-case delay of 16 clocks in a 64-length unit-stride vector access. With X-MP-2M protocol, this phenomena does not occur.

The relative importance of section renumbering to eliminate this effect is depicted in Figure 8a-b for CFD.2.32.2 and CFD.2.32.2M simulations. The former shows a cluster of delays between 13 and 16 clocks not present in the latter. The mean delay is reduced from 4.6 to 3.5 by section renumbering. Note that this effect is not evident in the DDF of CFD.16.256.2 of Figure 8c, even with the disadvantageous numbering. The reason is

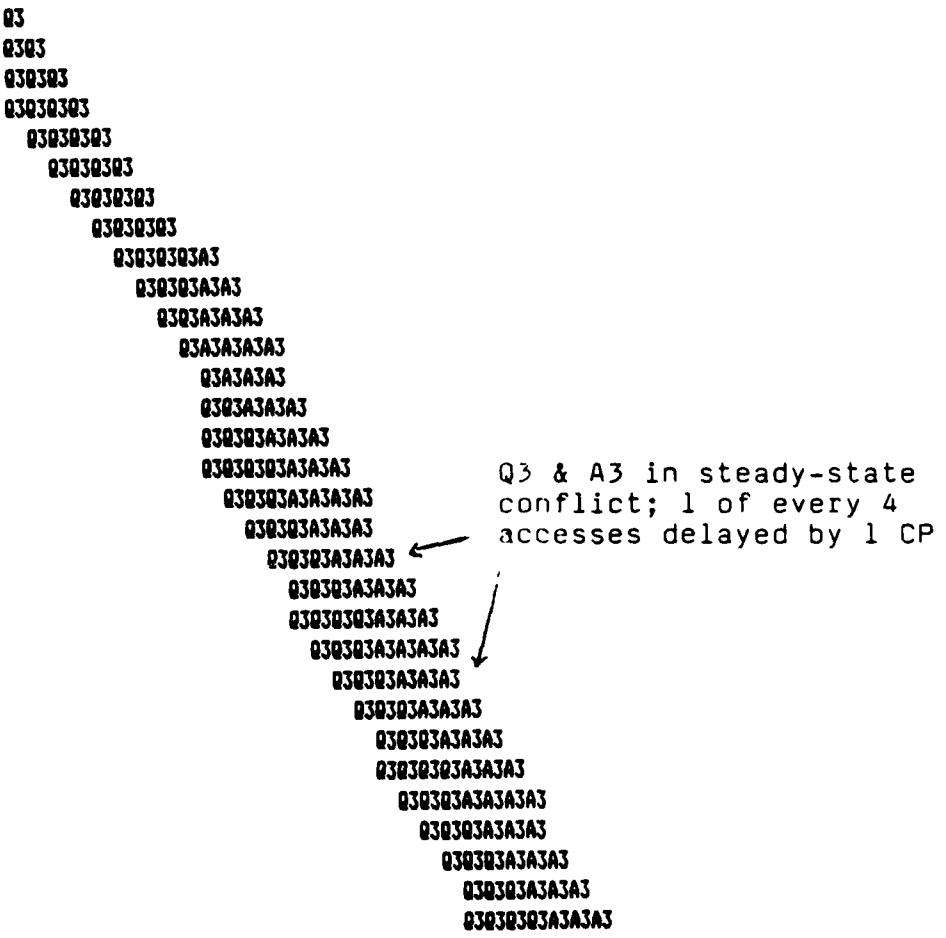
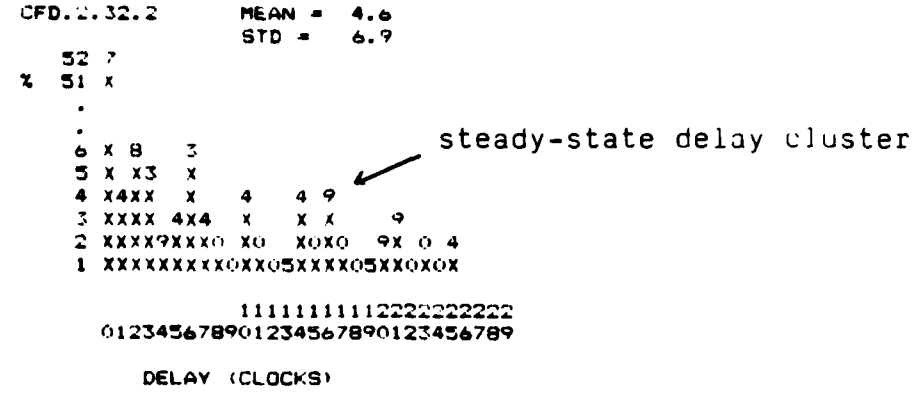
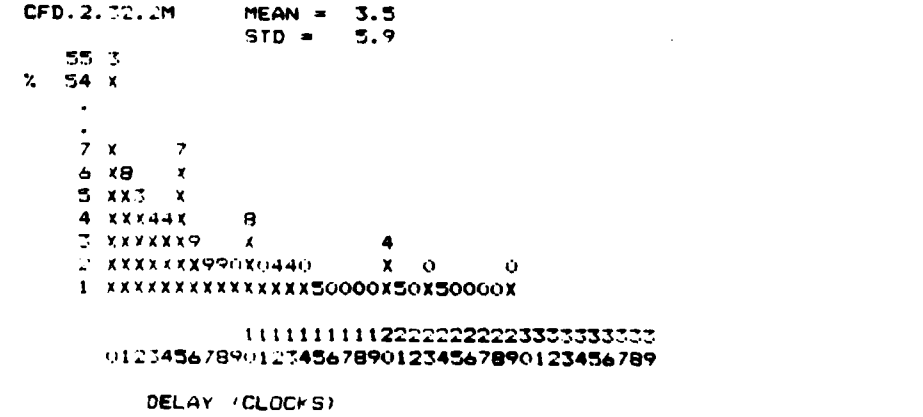


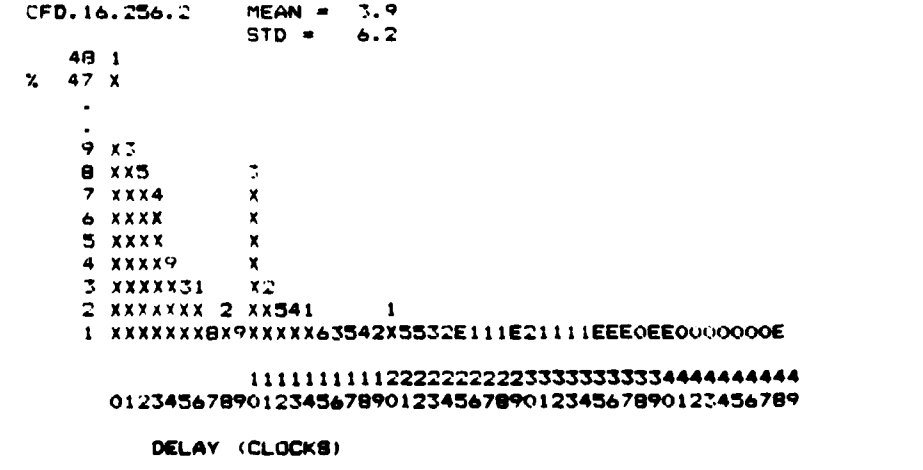
Figure 7. Illustration of steady-state conflict with X-MP-2 section numbering (also see [5]).



(a) X-MP-2; 2 processors



(b) X-MP-2M; 2 processors



(c) X-MP-2; 16 processors; 256 banks

Figure 8. Effect of steady-state conflict with different protocols and different number of processors.

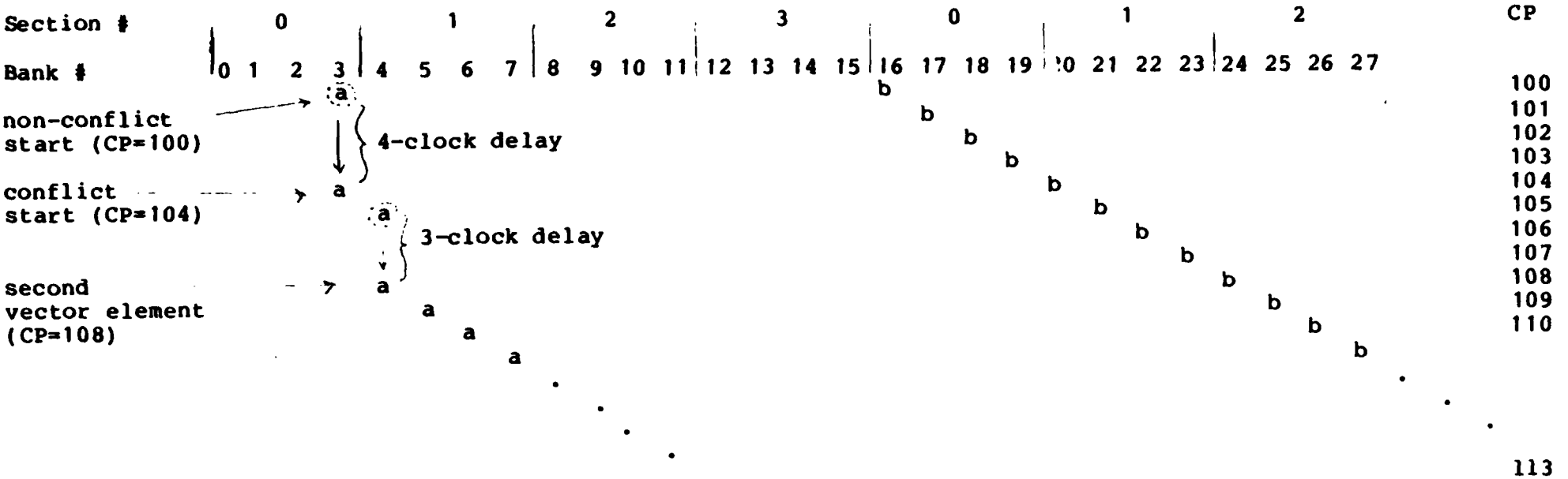
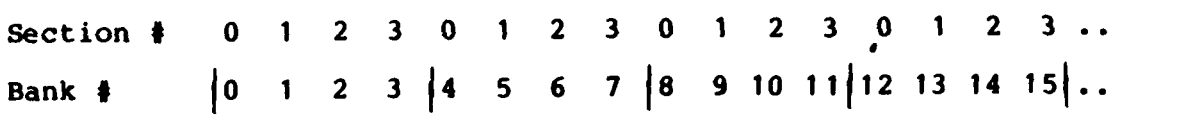
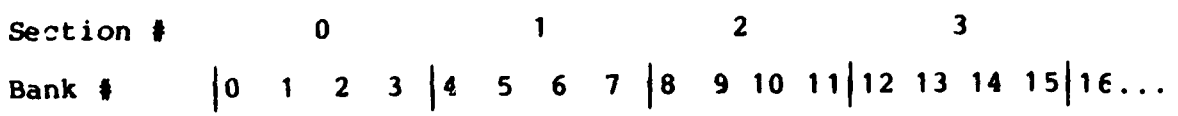


Figure 9a. Example of 7-clock startup delay in uniprocessor with X-MP-2M numbering.



(a) X-MP-2



(b) X-MP-4, X-MP-2M

Figure 9b. Relation between section and bank numbering

that the probability of two accesses being in the same neighborhood with 256 banks will in general be much less than with 32 banks. This effect is therefore significant only with a small number of banks.

c. Two-port startup section conflicts*

Although X-MP-2M protocol eliminates steady-state conflicts, it introduces a startup conflict phenomena not significant with X-MP-2 protocol. Figure 9a illustrates a worst case example of this conflict. Here, two ports of a processor are involved. Instruction b is in progress at CP 100 when instruction a attempts to access bank #3. b will have priority because it is in progress so a will hold until CP 104. Because both are in Section #0, a will then start a 4-clock access (only the first clock is marked in the figure); however, on CP105 it will again conflict with b, since both are now in Section #1, causing a three clock delay in a.

An instruction b in prioritized execution may be accessing bank 4r, 4r+1, 4r+2, or 4r+3 when instruction a attempts to startup in any of 16 banks in sections 0, 1, 2, or 3. There are therefore 4×16 = 64 distinctive relative startup positions of a and b. Each produces a startup delay, which may be counted and summed to produce an average startup delay. For the X-MP-4 section numbering, this number of startup delay clocks is 112, yielding an average delay

$$\bar{D}_{ac} = \frac{112}{64}$$
$$= 1.75$$

*Startup section conflict analysis assumes an infinite-bank memory, where only comflicts between periodic sections are accounted for; the probability of two accesses being made to the same bank is assumed zero.

clocks or 2.74% for VL=64. The corresponding delay for X-MP-2 numbering is

$$D_{ac} = .25$$

The same analysis which yielded the above D_{ac} can be used to evaluate the mean delay when one of two accesses, each in conflict-free steady state access, is bumped a prescribed number of clocks. Specifically, with b in prioritized execution as in Figure 9a, all 16 possible startup states of a are tested to determine which represent a conflict-free execution. For the valid states, the access of a is intentionally delayed (bumped) a prescribed number of clocks at CP100, to determine a new startup-like condition. This may result in a being in another conflict-free access, or a may now be in a conflict with b, and an extra delay incurred. These delays are summed and averaged over all possible valid states of a and b; the results are shown in Table 2. For example, an access bumped by 4 clocks would, on the average, incur a total delay of 4 + 2.44 = 6.44 clocks before it reached a new steady state compatible with instruction b. The X-MP-2 protocol would produce a total delay of 4.25 clocks.

This amplification of access bumps is felt to have a major role in the relatively poor performance of the X-MP-2M and X-MP-4. This analysis illustrates the potential for disruptive transients propogating across accesses and dramatically increasing delays. Analysis of such a dynamic situation is beyond the scope of this report.

d. Three-port startup section conflicts

The last section considered pairs of instructions representing a time when two ports are active. With two ports

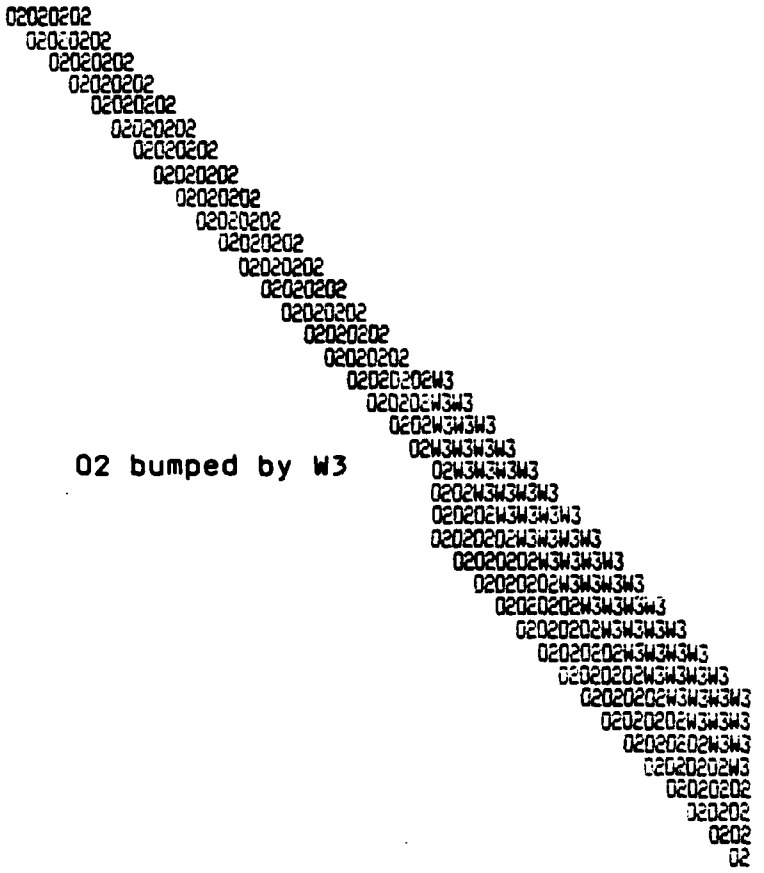


Figure 10. Memory utilization map illustrating a bumped conflict.

Bump (clocks)	Extra Delay X-MP-2 protocol (clocks)	X-MP-2M protocol (clocks)
0	0.	0.
1	.25	.778
2	.25	1.44
3	.25	2.00
4	.25	2.44
5	.25	2.78
6	.25	3.00
7	.25	3.11
8	.25	3.11
9	.25	3.11
10	.25	2.33
11	.25	1.67
12	.25	1.11
13	.25	.67
14	.25	.33
15	.25	.11
16	0.	0.

Table 2. Extra average delay suffered by a bumped access.

active and in the steady state, if the third port were to initiate an access, it is relatively easy to show that, depending on the relative location of the first two accesses, only 2-5 banks of every 16 can accommodate a third conflict-free startup (banks 27 and 28 in Figure 11).

To evaluate startup conflict with three ports it is possible to set two instructions (a and b) in a conflict-free steady-state mode, and then count the delays incurred by a third instruction c initiating an access in each of 16 banks. This is repeated for all combinations of a and b in a conflict-free steady state (36 rather than the 64 of the last section). A worst case example is illustrated in Figure 11, where a 14-clock delay is indicated. Overall, among 36×16 = 576 cases, a total of 2944 delay clocks are counted, for

$$D_{ac} = \frac{2944}{576}$$

$$= 5.11$$

clocks average startup delay.

4. Effects of Section Design Parameters

The above startup delay analyses for two-port and three-port accesses with X-MP-2M numbering can be performed as a function of the number of sections and the number of banks per section - both equal to 4 in the above study.

Table 3 gives the results of enumerating all combinations of instruction startups and averaging delays, as above. Three results are worthy of note.

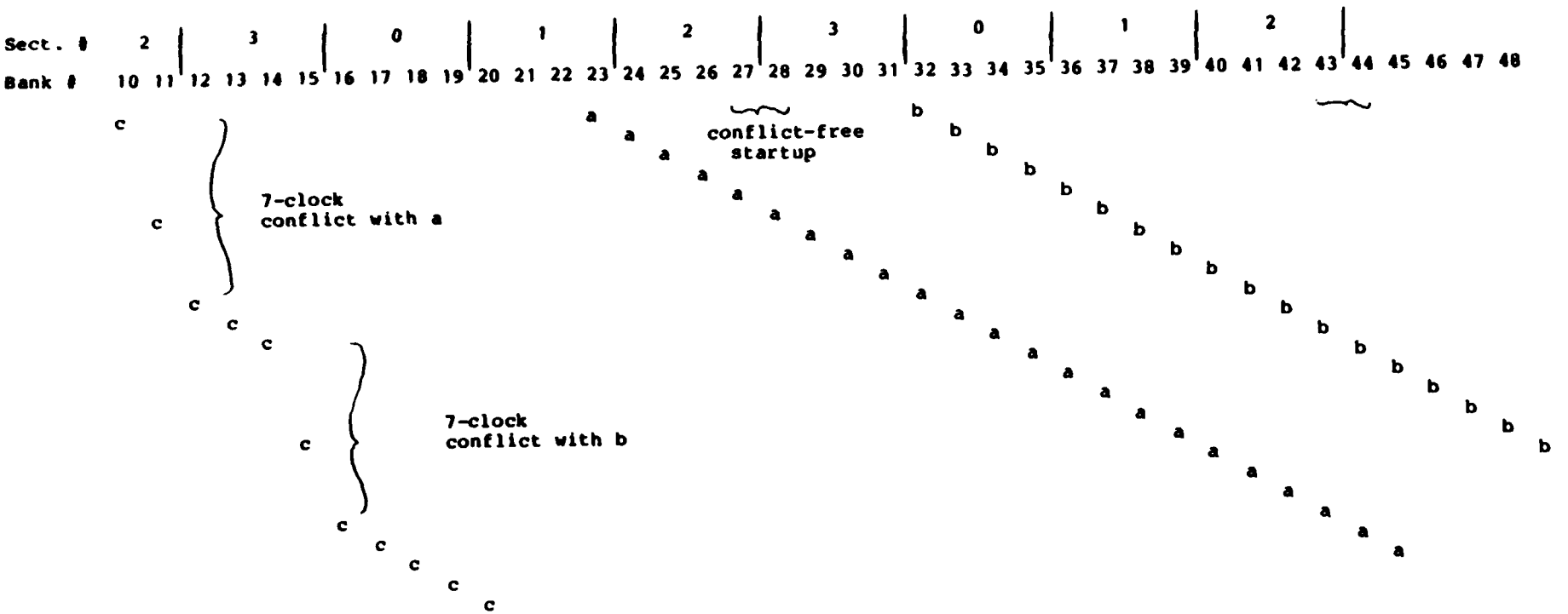


Figure 11. Worst case startup delay with three active ports; a and b in prioritized progress.

NBPS	NS	2-port access (clocks)	3-port access (clocks)
2	2	1.5	---
	4	.75	4.2
	8	.38	.86
4	2	3.50	---
	4	1.75	5.11
	8	.88	2.04
8	2	7.50	---
	4	3.75	11.2
	8	1.88	4.40
	16	.94	1.93
16	2	15.5	---
	4	7.75	23.5
	8	3.87	9.11
	16	1.93	4.15

Table 3. Startup section delays as function of the number of sections (NS and the number of banks per section (NBPS).

(a) If the ratio

$$R_{bs} = \frac{\text{\# of sections (NS)}}{\text{\# of banks per section (NBPS)}}$$

is maintained constant, both the two-port and three-port startups are relatively constant. If $R_{bs} = 1$, for example, the three-port startups of 5.11, 4.40 and 4.15 clocks are determined.

(b) For a given NBPS, the delay decreases as the inverse in the increase in NS. This is reasonable, since no delays are encountered for additional sections.

(c) For a given NS, the delay increases proportionately to the increase in NBPS. This is explained by the doubling of the number of delay clocks when an instruction enters a reserved double-width section.

5. Startup bank conflicts

In codes without section conflicts (e.g., one-port codes), the only possible conflicts are <u>startup</u> bank conflicts between processors. In general codes, such conflicts form a low-level conflict background which interacts principally with section startup conflicts and buffer fetches.

Define the memory utilization

$$\bar{U}_m = \frac{\text{total memory accesses}}{\text{total run time}}$$

for a uniprocessor. Each access occupies a 4-clock wide path in the memory utilization map. With NB banks, $\bar{U}_m$ memory utilization per processor, and p processors, the fraction of total bank-width occupied by accesses is approximately $P_d = \bar{U}_m/R_{bp}$. For three clocks on either side of these accesses, a startup will either be delayed or will force a delay in the existing access. The probabilities associated with various delays are

The probabilities associated with various delays are

$$\text{pr (3-clock delay)} = P_d$$

$$\text{pr (3-clock delay)} = 2P_d$$

$$\text{pr (2-clock delay)} = 2P_d$$

$$\text{pr (1-clock delay)} = 2P_d$$

The average startup delay due to bank conflicts is computed to be

$$\bar{D}_{acbc} = \frac{16}{7} \bar{U}_m / R_{bp}.$$

For typical values of R_{bp} (=8,16), this delay is significantly less than one clock. It should be noted that "one port" codes - which have no section conflicts - can have a highly regular bank conflict pattern. A version of MUL_3, which utilizes more than one port only 11% of the time and has a regular access pattern, produced the DDF pattern of Figure 12d. Note that the frequency of 4-clock and 8-clock delays dominate the frequency of other non-zero delays. Clearly, the equal-probability assumptions of the above analysis are inappropriate. Section delays appear to have a randomizing, albeit negative, effect.

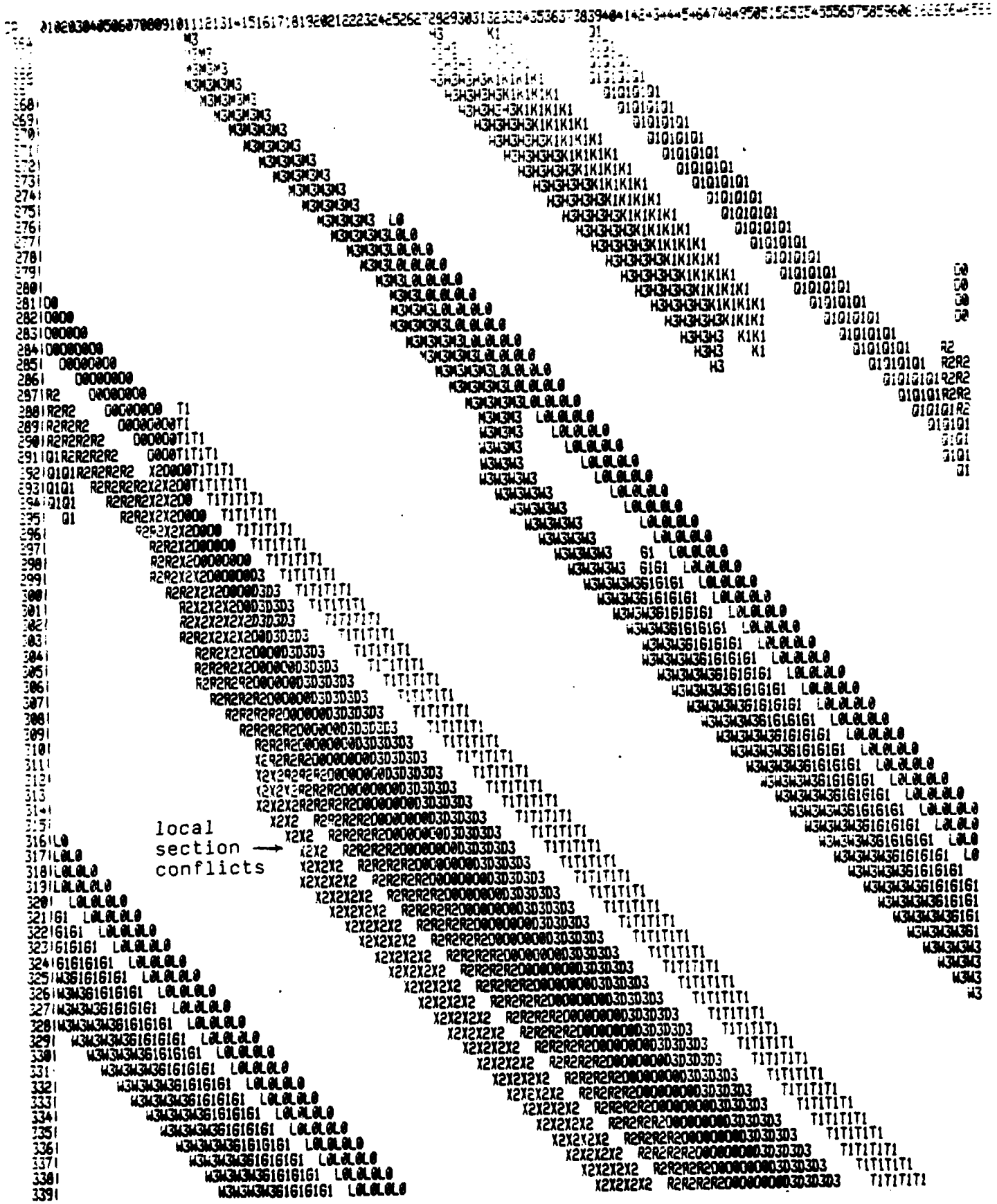


Figure 12a. Memory utilization map with X-MP-2 protocol

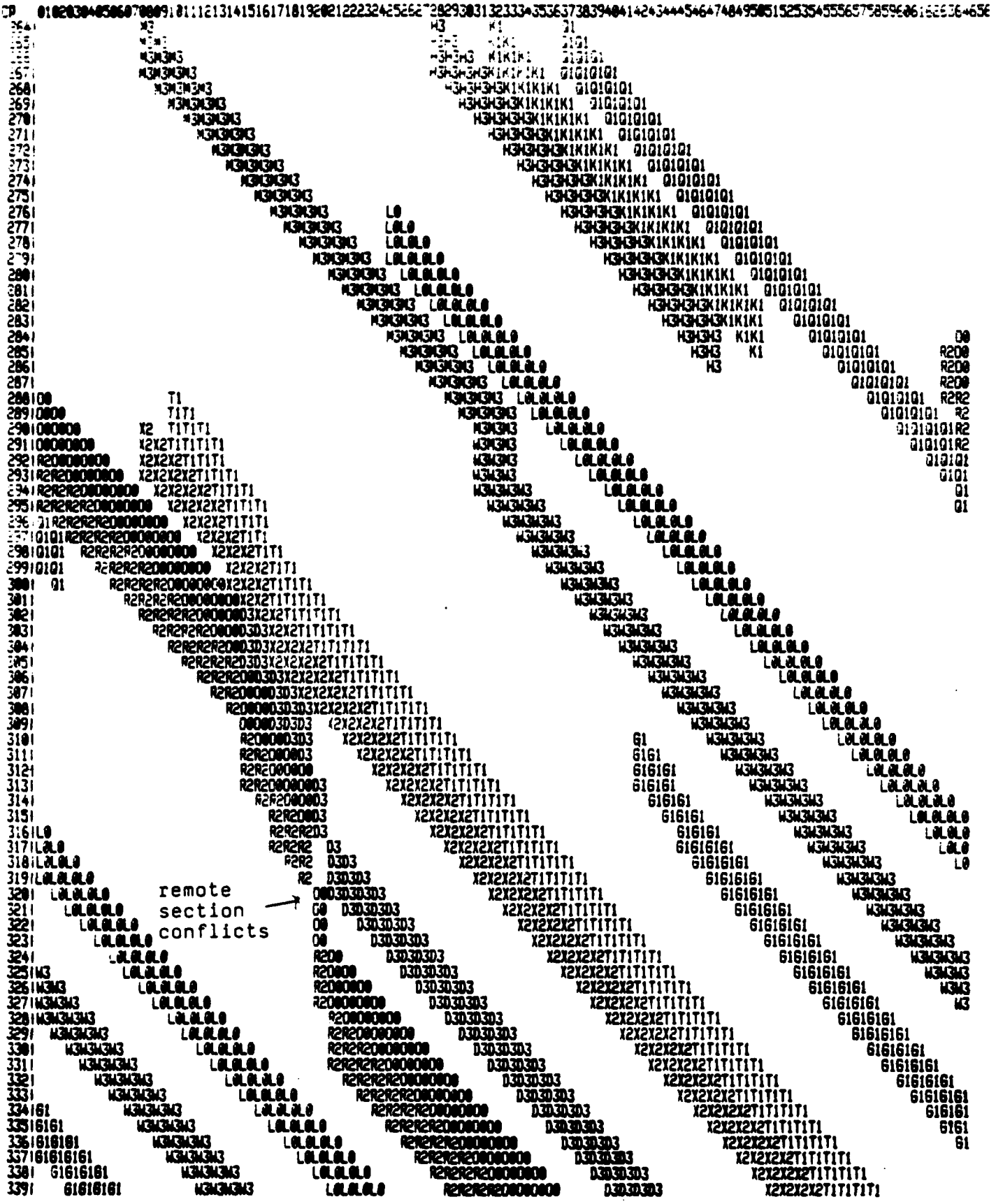


Figure 12b. Memory utilization map with X-MP-2M protocol

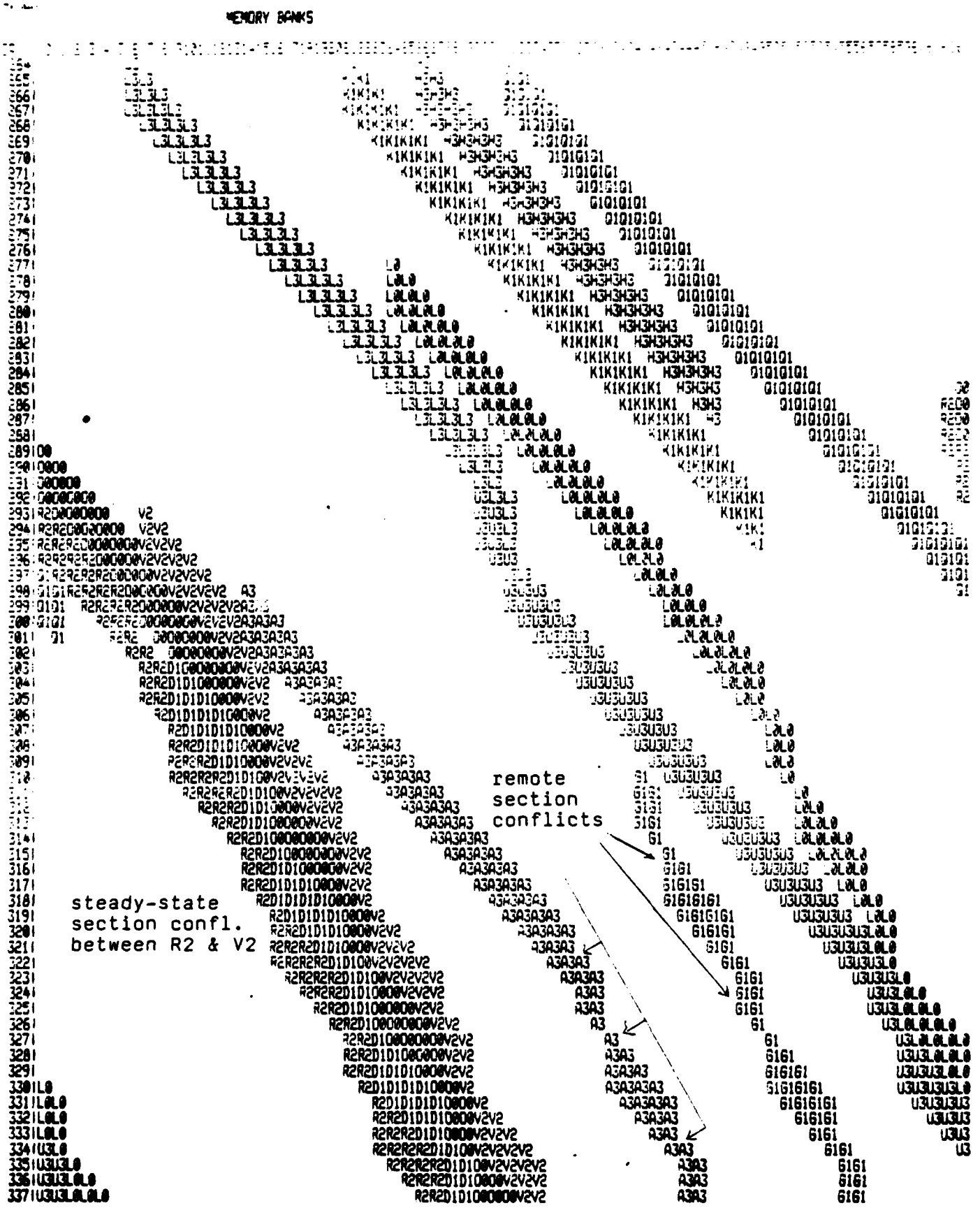


Figure 12c. Memory utilization map with X-MP-4 protocol

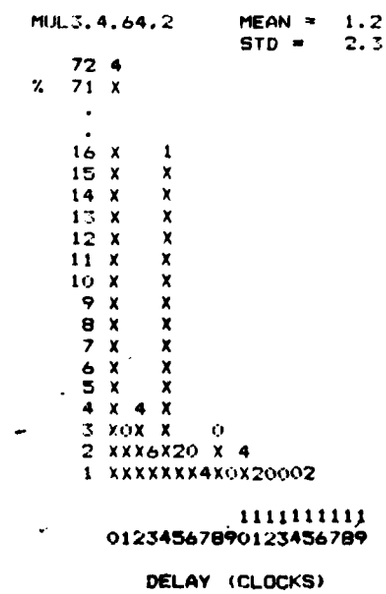


Figure 12d. Granular DDF for one-port multiply code; 4 processors.

IV. EVALUATION OF CONFLICT RESOLUTION PROTOCOLS

A. INTRODUCTION

B. AGGRAGATE PERFORMANCE

This section considers the aggragate performance characteristics associated with the three protocols simulated, using delays averaged across all six codes.

Figure 13 and Table 4 compare delays associated with three protocols as a function of the number of processors. In all cases, $R_{bp} = 16$; thus, characteristics displayed as a function of p could as well be shown as a function of NB.

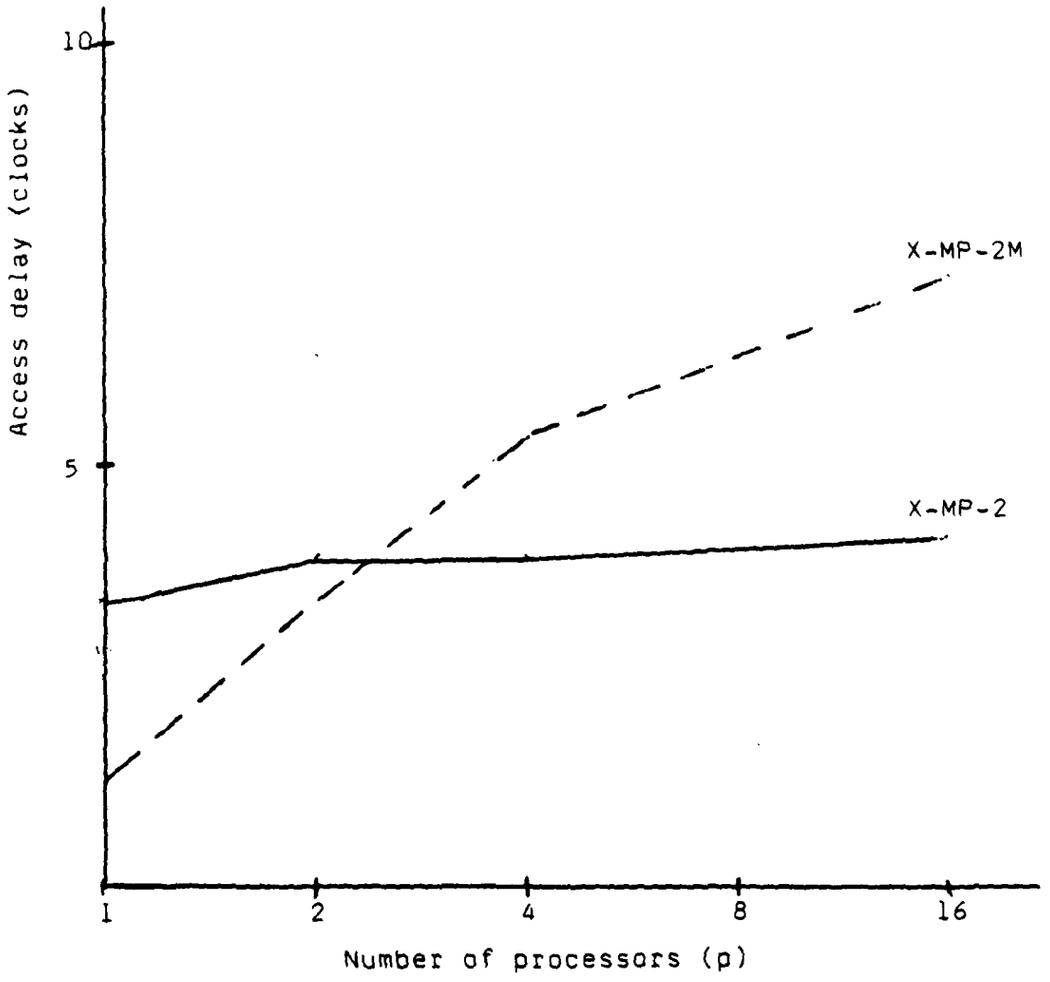
The effect of section numbering is highlighted in this comparison. Because the X-MP-2M protocol avoids the steady-state section conflict associated with neighboring accesses from the same processor, Figure 13 shows X-MP-2M protocol is favored for p = 2. This relative advantage of X-MP-2M decreases as a function of NB when the likelihood of neighboring access decreases. Indeed, the section startup disadvantage of the X-MP-2M discussed in section III.E.3.c-d begins to dominate for p > 2; for p = 16 (NB = 256), the delays of the X-MP-2M is 71% greater than that of the X-MP-2.

The continued increase in Figure 13 of access delay for p > 4 for X-MP-2M protocol is not predictable by the theory of this report. It is surmised that, as p increases, the accesses become less patterned and this randomness increases the collision frequency. This slope would also appear with the X-MP-2 characteristic if the abovementioned steady-state delays had not increased the total delay for small p; that is, the steady-state delays decrease with p while the startup delays increase with p, giving a combined flat X-MP-2 characteristic.

Figure 13

Composite access delays of test codes

R_{bp} = 16 ; VL = 64



Code	X-MP-2	X-MP-2M
1-processor		
MUL_2	2.4	1.7
MUL_1	2.7	1.7
MUL_3	2.2	.3
RAN	6.0	2.1
FFT	2.8	.9
CFD	4.2	1.3
Average	3.4	1.3
2-processor		
MUL_2	2.9	4.5
MUL_1	2.0	2.7
MUL_3	4.0	1.5
RAN	5.9	4.3
FFT	3.1	3.3
CFD	4.6	3.5
Average	3.8	3.3
4-processor		
MUL_2	3.6	7.6
MUL_1	3.9	5.9
MUL_3	1.2	1.2
RAN	6.5	7.3
FFT	3.5	5.1
CFD	3.8	4.6
Average	3.8	5.3
16-processor		
MUL_2	3.8	10.6
MUL_1	3.2	7.0
MUL_3	1.7	2.6
RAN	7.6	10.9
FFT	4.9	6.6
CFD	3.9	5.2
Average	4.2	7.2

Table 4. Averaged access delays (clocks) for six codes with three bank conflict protocols. R_{pb}=16.

C. DEPENDENCE OF D_{ac} ON R_{bp}

Figure 13 and Table 2 indicate what may be regarded as unacceptable delays $\bar{D}_{ac}$ with X-MP-2M protocol, especially for 16 processors. It is possible to consider reducing $\bar{D}_{ac}$ by using more memory banks; in this case, the question becomes the dependence of $\bar{D}_{ac}$ on R_{bp}, when $R_{bp} > 16$.

Table 5 shows the dependence of the two codes with the largest $\bar{U}_m$ (and the largest $\bar{D}_{ac}$ in Table 3) on N_b with $p = 4$, with X-MP-4 protocol. For $\bar{D}_{ac}$'s which may be objectionable (>10 clocks), the delay is decreased by a factor of 2.5-3.1 by doubling the number of banks. Further increase in N_b has marginal benefit.

	$\bar{D}_{ac}$ (clocks)			
Code	N_b=32	N_b=64	N_b=128	N_b=256
4 processors				
MUL_2	21.9	10.3	3.7	2.5
RAN^2	--	18.9	7.5	4.5
8 processors				
MUL_2	--	--	15.2	5.2
RAN^2	--	--	21.3	8.4
16 processors				
MUL_2	--	--	--	20.5
RAN^2	--	--	--	21.9

Table 5. Effect of increasing number of banks

REFERENCES

[1] Chen, S., J. Dongarra, and C. Hsuing, "Multiprocessing Linear Algebra Algorithms on the CRAY X-MP-2: Experiences with Small Granularity," Mathematics and Computer Science Division Technical Memorandum No. 24, Argonne National Laboratory, February, 1984.

[2] Moore, M., R. Hiromoto, and O. Lubeck, "Experiences with the Denelsor HEP," to appear in Parallel Computing, North Holland Publisher.

[3] Axelrod, T.S., P. F. Dubois, and P. Eltgroth, "A Simulator for MIMD Performance Prediction--Application of the S-1 MkIIa Multiprocessor," Report UCAL-88765, Lawrence Livermore National Laboratory, February, 1983.

[4] Calahan, D.A., "Influence of Task Granularity on Vector Multiprocessor Performance," Proc. 1984 Intl. Conf. on Parallel Processing, Bellaire, MI, August 21-24, 1984; pp 278-284.

[5] Cheung, Tony, and J. E. Smith, "An Analysis of the CRAY X-MP Memory System," Proc. 1984 Intl. Conf. on Parallel Processing, Bellaire, MI, August 21-24, 1984; pp 494-505.

[6] Cray Research, Inc., "Cray X-MP Series Mainframe Reference Manual," HR-0032, Nov. 1982.

[7] Buning, P.G., and J. B. Levy, "Vectorization of Implicit Navier-Stokes Codes on the CRAY-1 Computer," Dept. of Aeronautics and Astronautics, Stanford University, November 15, 1979.

[8] Calahan, D.A., "Conflict Sensitivity of Algorithms. Part I: A CRAY X-MP Study," Report SARL #7, Dept. of Elec. Engr. and Comp. Sci., University of Michigan, March, 1985.

APPENDIX A

EXPERIMENT DESCRIPTION

EXPERIMENTAL PARAMETERS

The codes were produced by the X-MP CFT compiler from Fortran source codes. Vector length (VL) is 64 and stride is 1 for all cases.

Distinct program and data storage was used for each of the 16 processors. Code executions were initiated at irregular intervals to further randomize accesses between processors. In general, p samples were used to produce mean values with p processors.

Two global static measures of memory accesses were made to monitor their uniformity.

(a) Memory utilization. This is the fraction

$$\bar{U}_m = \frac{\text{Total operands and results}}{\text{Simulation time (CP's)}}$$

for the average processor; it is a measure of memory traffic for each code, and has a maximum value of 3, corresponding to the number of memory ports per processor. Table 1 shows $U_m \approx .67$ for FFT, CFD, and MUL_1.

(b) Bank utilization. Let N_b be the number of banks. There is a risk with 64-length unit-stride vectors and $N_b > 64$ that banks will not be equally utilized; this would create uncharacteristic delays in heavily-utilized banks. If $\bar{N}$ is the average number of accesses per processor across all banks, and N is the standard deviation from this average, define the bank utilization

$$\bar{U}_b = \frac{\bar{N} - N}{\bar{N}}.$$

$\bar{U}_b = 1$ indicates uniform accessing; if only 1/2 of the banks are accessed, $\bar{U}_b = 1/2$. Table 1 indicates $.832 < \bar{U}_b < .998$.

CODE DESCRIPTIONS

(a) Fluids kernel (CFD). Taken from the vectorized code of [7], this a 32-statement single-loop Fortran kernel with an average of 3.2 64-length vector-vector operations/statement. Lack of a repetitive computational structure like FFT and MUL should make the access pattern the most random. Six buffer fetches occur in one kernel execution.

(b) FFT kernel (FFT). This code determines multiple 8-point complex-complex FFT's. Five buffer fetches occur in one kernel execution.

(c) Matrix-vector multiply kernel (MUL_1, MUL_2, MUL_3). The inner-loop of MUL_1 and MUL_2 has two vector reads and one write per execution. MUL_1 maintains low memory utilization ($U_m = .69$) with VL = 64 by multiplying 4 small (64 x 3) matrices in one kernel execution step; MUL_2 uses the same code with 512×2 matrices, which successively exercises the inner-loop 512/64 = 8 times, and achieves $U_m = 1.58$, a value more characteristic of a large Fortran-coded matrix multiply on the X-MP. No buffer fetches occur in consecutive executions of the kernel. The inner loop of MUL_3 has one pre-fetched vector read per inner loop execution.

Code	Memory Utilization $\overline{U_m}$	Bank Utilization $\overline{U_b}$	$\overline{AP}^x$ (% of time) 0	1	2	3	Conflicts+ $\overline{BC}$	$\overline{SC}$
RAN	1.62	NR*	3.7	29.8	66.5	0	NR*	NR
FFT	.653	.965	47.1	35.7	9.7	7.5	.908	.141
CFD	.682	.986	49.3	31.9	15.6	3.2	.469	.031
MUL_2	1.53	.998	22.8	24.8	28.9	23.5	1.55	.343
MUL_1	.702	.832	62.3	14.7	13.4	9.6	.840	.135
MUL_3	.932	.996	16.8	73.1	11.1	0	.534	.036

$^x\overline{AP}$ - average active ports

*NR - not recorded

$^+\overline{BC}$ - average bank conflicts per clock

$\overline{SC}$ - average section conflicts per clock

$^y\overline{BFD}$ - average buffer fetch delay per access

Table A. Code characterization; 16 processors, 256 banks, X-MP-2 protocol.

APPENDIX B

SIMULATOR VALIDATION

A. INTRODUCTION

The X-MP simulator was validated by comparison with a 16-bank X-MP-2.

Our experience with a CRAY-1 simulator indicates that we can expect to achieve a timing accuracy within 1% for typical kernels, without bank conflicts. However, since the purpose of this report is to study memory design parameter dependence on greatly-extrapolated architectures in high-conflict situations, credibility required closer validation.

In particular , it was felt necessary to validate conflicting memory reference timing more precisely. Consequently, three types of validations were made:

(1) Clock-level accuracy was tested for short runs of high-conflict memory reads and writes.

(2) Statistical validation was made for long runs of very-high-conflict reads and writes.

(3) Overall instruction timing accuracy was checked with a low-conflict linear algebra code.

B. CLOCK-LEVEL VALIDATION

1. Effect of priority switch on validation.

The X-MP-2 and X-MP-4 establish priority for simultaneous bank conflicts between p processors with a rotating priority queue that changes state every four clocks. This results in potentially 4p different timings for each multiprocessor run, corresponding to

4p initial states of the queue. The relation of the queue state to the real time clock is fixed at hardware startup, but may change as a result of shutdown; the state of the switch cannot be directly monitored.

When clock-level accuracy was to be tested, each code executed m(4p) times, where m > 4 to insure reproducabiity. Each of the m runs was started at the same queue state, determined by masking the real time clock in a loop before the code was entered; the loop was exited only when a desired mask was obtained, and the loop length was chosen to advance the mask one clock upon each loop execution. The next m runs of the code were made by advancing the desired mask.

2. Clock-level validation of read/write tests.

Two codes consisting of vector reads and writes with random bank starting address, random stride (< 64), and random vector length (< 64) were synchronized at the clock level and left to run for several hundred clocks (until a buffer fetch occurred) on an X-MP-2. Termination times were recorded for each of the eight initial states; in this period of time, approximately 2000 bank conflicts and 300 section conflicts were recorded. Simulated and actual run times matched precisely for all cases, after a phase adjustment associated with the hardware indetermancy of the priority queue relative to the real time clock.

3. Statistical validation of read/write tests.

In this test, processor #1 (P1) issued a series of 64-length, unit-stride vector reads against a background of random reads and

writes (see above) in a second processor (P2). The time of the read-path reservation in P1 was recorded, and the mean and standard deviation of this time determined. This data was collected for up to 64000 reads in P1 on the hardware to determine the dependence of the statistical results on the number of trials.

Without bank conflicts, this reservation time is 69 clocks. The high-conflict nature of the code in P2 is evidenced by a 107-clock mean delay (55% overhead) for both hardware and simulated timings. This validates the long-term read/write simulator performance for even high-conflict cases.

4. Overall instruction timing validation.

A low-conflict CAL LU uniprocessor factorization referenced in [1] [2] was compared with simulated performance. A .27% error in simulated performance occurred with bank conflicts.

END

FILMED

11-85

DTIC